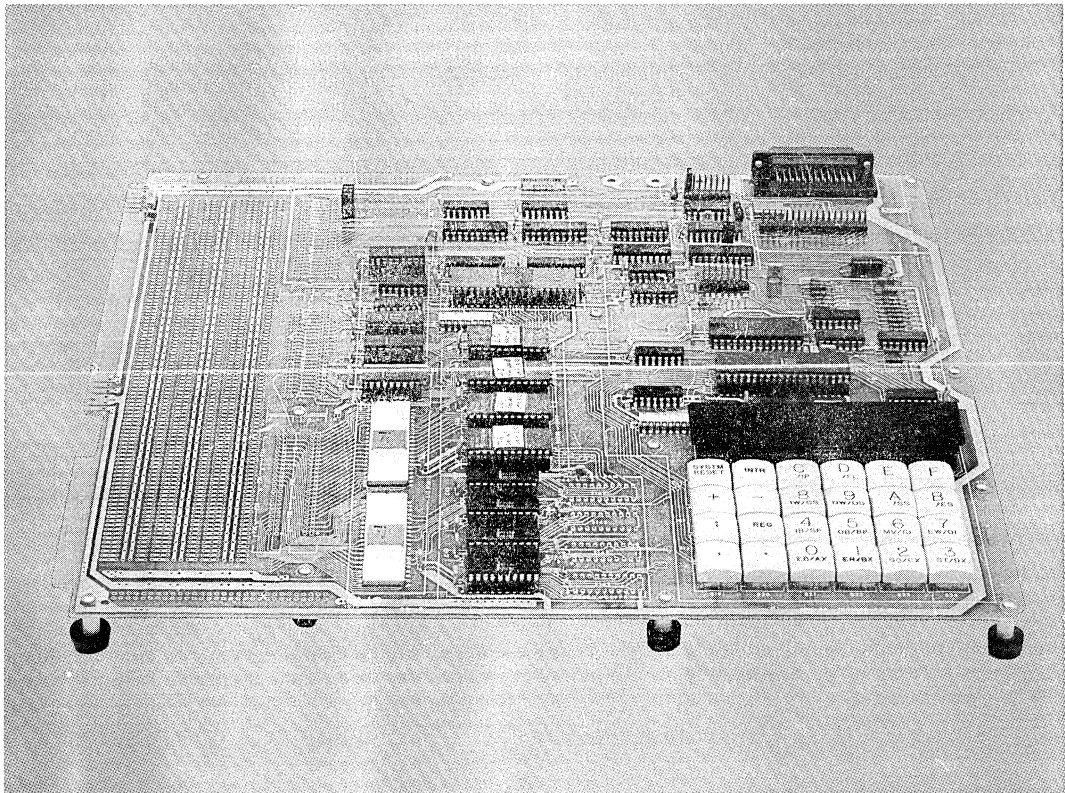




SDK-86 MCS-86™ SYSTEM DESIGN KIT

- Complete Single Board Microcomputer System Including CPU, Memory, and I/O
- Easy to Assemble Kit Form
- High Performance 8086 16-Bit CPU
- Interfaces Directly with TTY or CRT
- Interactive LED Display and Keyboard
- Wire Wrap Area for Custom Interfaces
- Extensive System Monitor Software in ROM
- Comprehensive Design Library Included

The SDK-86 MCS-86 System Design Kit is a complete single board 8086 microcomputer system in kit form. It contains all necessary components to complete construction of the kit, including LED display, keyboard, resistors, caps, crystal, and miscellaneous hardware. Included are preprogrammed ROMs containing a system monitor for general software utilities and system diagnostics. The complete kit includes an 8-digit LED display and a mnemonic 24-key keyboard for direct insertion, examination, and execution of a user's program. In addition, it can be directly interfaced with a teletype terminal, CRT terminal, or the serial port of an Intellec system. The SDK-86 is a high performance prototype system with designed-in flexibility for simple interface to the user's application.



FUNCTIONAL DESCRIPTION

The SDK-86 is a complete MCS-86 microcomputer system on a single board, in kit form. It contains all necessary components to build a useful, functional system. Such items as resistors, caps, and sockets are included. Assembly time varies from 4 to 10 hours, depending on the skill of the user. The SDK-86 functional block diagram is shown in Figure 1.

8086 Processor

The SDK-86 is designed around Intel's 8086 microprocessor. The Intel 8086 is a new generation, high performance microprocessor implemented in N-channel, depletion load, silicon gate technology (HMOS), and packaged in a 40-pin CerDIP package. The processor features attributes of both 8-bit and 16-bit microprocessors in that it addresses memory as a sequence of 8-bit bytes, but has a 16-bit wide physical path to memory for high performance. Additional features of the 8086 include the following:

- Direct addressing capability to one megabyte of memory
- Assembly language compatibility with 8080/8085
- 14 word \times 16-bit register set with symmetrical operations
- 24 operand addressing modes
- Bit, byte, word, and block operations
- 8 and 16-byte signed and unsigned arithmetic in binary or decimal mode, including multiply and divide
- 4 or 5 or 8 MHz clock rate

A block diagram of the 8086 microprocessor is shown in Figure 2.

System Monitor

A compact but powerful system monitor is supplied with the SDK-86 to provide general software utilities and system diagnostics. It comes in preprogrammed read only memories (ROMs).

Communications Interface

The SDK-86 communicates with the outside world through either the on-board light emitting diode (LED) display/keyboard combination or the user's TTY or CRT terminal (jumper selectable), or by means of a special mode in which an Intellec development system transports finished programs to and from the SDK-86. Memory may be easily expanded by simply soldering in additional devices in locations provided for this purpose. A large area of the board (22 square inches) is laid out as general purpose wire-wrap for the user's custom interfaces.

Assembly

Only a few simple tools are required for assembly: soldering iron, cutters, screwdriver, etc. The SDK-86 assembly manual contains step-by-step instructions for easy assembly with a minimum of mistakes. Once construction is complete, the user connects his kit to a power supply and the SDK-86 is ready to go. The monitor starts immediately upon power-on or reset.

Commands — Keyboard mode commands, serial port commands, and Intellec slave mode commands are summarized in Table 1, Table 2, and Table 3, respectively. The SDK-86 keyboard is shown in Figure 3.

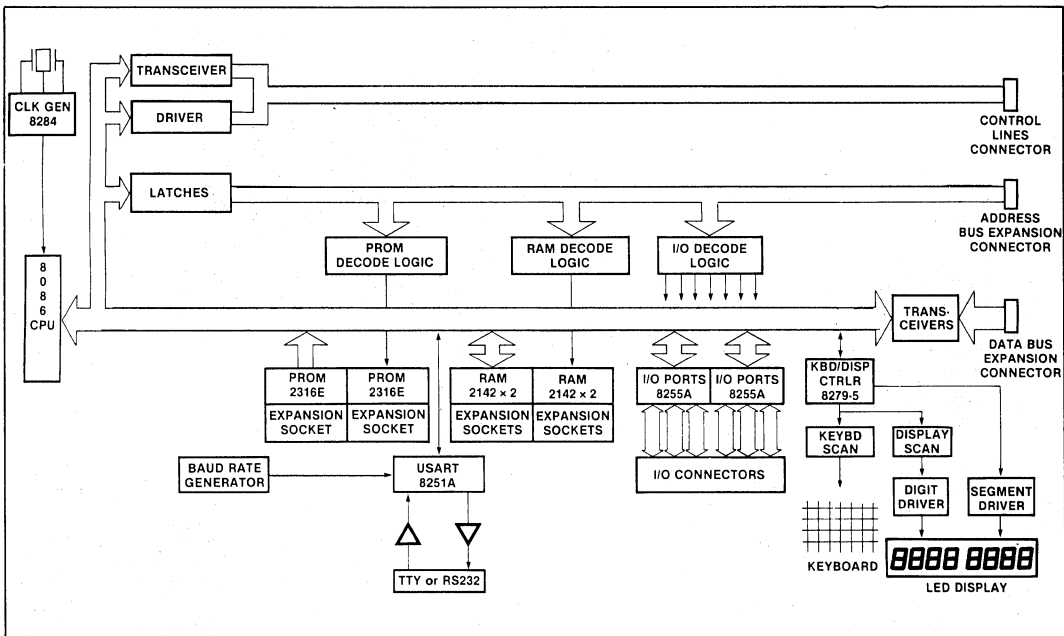


Figure 1. SDK-86 System Design Kit Functional Block Diagram

Documentation

In addition to detailed information on using the monitors, the SDK-86 user's manual provides circuit diagrams, a monitor listing, and a description of how the system works. The complete design library for the SDK-86 is shown in Figure 4 and listed in the specifications section under Reference Manuals.

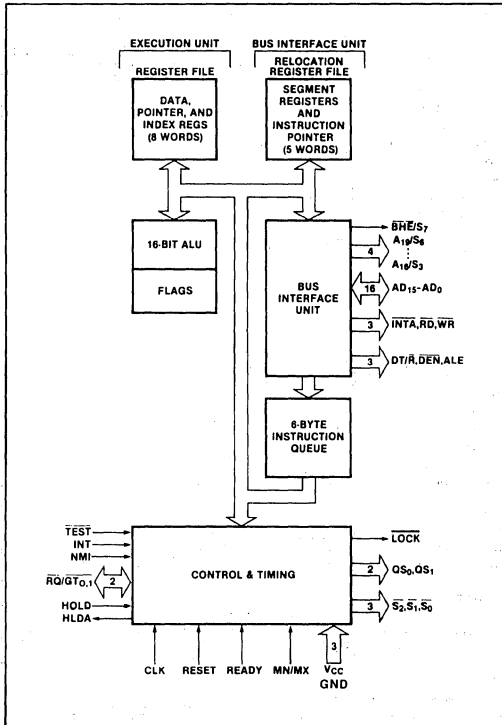


Figure 2. 8086 Microprocessor Block Diagram

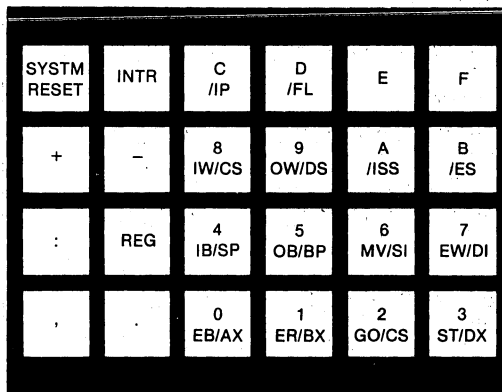


Figure 3. SDK-86 Keyboard



Figure 4. SDK-86 Design Library

Table 1. Keyboard Mode Commands

Command	Operation
Reset	Starts monitor.
Go	Allows user to execute user program, and causes it to halt at predetermined program stop. Useful for debugging.
Single step	Allows user to execute user program one instruction at a time. Useful for debugging.
Substitute memory	Allows user to examine and modify memory locations in byte or word mode.
Examine register	Allows user to examine and modify 8086 register contents.
Block move	Allows user to relocate program and data portions in memory.
Input or output	Allows direct control of SDK-86 I/O facilities in byte or mode.

Table 2. Serial Mode Commands

Command	Operation
Dump memory	Allows user to print or display large blocks of memory information in hex format than amount visible on terminal's CRT display.
Start/continue display	Allows user to display blocks of memory information larger than amount visible on terminal's CRT display.
Punch/read paper tape	Allows user to transmit finished programs into and out of SDK-86 via TTY paper tape punch.

8086 INSTRUCTION SET

Table 4 contains a summary of processor instructions used for the 8086 microprocessor.

Table 4. 8086 Instruction Set Summary

Mnemonic and Description	Instruction Code				Mnemonic and Description	Instruction Code			
Data Transfer					CMP - Compare:				
MOV - Move:	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	Register/memory and register	0 0 1 1 1 0 d w	mod reg	r/m	
Register/memory to/from register	1 0 0 0 1 0 d w	mod reg	r/m		Immediate with register/memory	1 0 0 0 0 s w	mod 1 1 1 r/m	data	data if s/w=01
Immediate to register/memory	1 1 0 0 0 1 1 w	mod 0 0 0 r/m	data	data if w=1	Immediate with accumulator	0 0 1 1 1 0 w	data	data if w=1	
Immediate to register	1 0 1 1 w	reg	data	data if w=1	AAS-ASCII adjust for subtract	0 0 1 1 1 1 1			
Memory to accumulator	1 0 1 0 0 0 w	addr-low	addr-high		DAS-Decimal adjust for subtract	0 0 1 0 1 1 1 1			
Accumulator to memory	1 0 1 0 0 0 1 w	addr-low	addr-high		MUL-Multiply (unsigned)	1 1 1 1 0 1 1 w	mod 1 0 0 r/m		
Register/memory to segment register	1 0 0 0 1 1 1 0	mod 0 reg	r/m		IMUL-Integer multiply (signed)	1 1 1 1 0 1 1 w	mod 1 0 1 r/m		
Segment register to register/memory	1 0 0 0 1 1 0 0	mod 0 reg	r/m		AAM-ASCII adjust for multiply	1 1 0 1 0 1 0 0	0 0 0 0 1 0 1 0		
					DIV-Divide (unsigned)	1 1 1 1 0 1 1 w	mod 1 1 0 r/m		
PUSH - Push:					IDIV-Integer divide (signed)	1 1 1 1 0 1 1 w	mod 1 1 1 r/m		
Register/memory	1 1 1 1 1 1 1 1	mod 1 1 0 r/m			AAD-ASCII adjust for divide	1 1 0 1 0 1 0 1	0 0 0 0 1 0 1 0		
Register	0 1 0 1 0 reg				CBW-Convert byte to word	1 0 0 1 1 0 0 0			
Segment register	0 0 0 reg 1 1 0				CWD-Convert word to double word	1 0 0 1 1 0 0 1			
POP - Pop:									
Register/memory	1 0 0 0 1 1 1 1	mod 0 0 0 r/m							
Register	0 1 0 1 1 reg								
Segment register	0 0 0 reg 1 1 1								
XCHG - Exchange:					Logic				
Register/memory with register	1 0 0 0 0 1 1 w	mod reg	r/m		NOT-Invert	1 1 1 1 0 1 1 w	mod 0 1 0 r/m		
Register with accumulator	1 0 0 1 0 reg				SHL/SAL-Shift logical/arithmetic left	1 1 0 1 0 0 v w	mod 1 0 0 r/m		
IN - Input					SHR-Shift logical right	1 1 0 1 0 0 v w	mod 1 0 1 r/m		
Fixed port	1 1 1 0 0 1 0 w	port			SAR-Shift arithmetic right	1 1 0 1 0 0 v w	mod 1 1 1 r/m		
Variable port	1 1 1 0 1 1 0 w				RCL-Rotate left	1 1 0 1 0 0 v w	mod 0 0 0 r/m		
OUT - Output					ROR-Rotate right	1 1 0 1 0 0 v w	mod 0 0 1 r/m		
Fixed port	1 1 1 0 0 1 1 w	port			RCL-Rotate through carry flag left	1 1 0 1 0 0 v w	mod 0 1 0 r/m		
Variable port	1 1 1 0 1 1 1 w				RCR-Rotate through carry right	1 1 0 1 0 0 v w	mod 0 1 1 r/m		
XLAT-Translate byte to AL	1 1 0 1 0 1 1 1								
LEA-Load EA to register	1 0 0 0 1 1 0 1	mod reg	r/m		AND - And:				
LDS-Load pointer to DS	1 1 0 0 0 1 0 1	mod reg	r/m		Reg./memory and register to either	0 0 1 0 0 0 d w	mod reg	r/m	
LES-Load pointer to ES	1 1 0 0 0 1 0 0	mod reg	r/m		Immediate to register/memory	1 0 0 0 0 0 w	mod 1 0 0 r/m	data	data if w=1
LAHF-Load AH with flags	1 0 0 1 1 1 1 1				Immediate to accumulator	0 0 1 0 0 1 0 w	data	data if w=1	
STAHF-Store AH into flags	1 0 0 1 1 1 1 0								
PUSHF-Push flags	1 0 0 1 1 1 0 0				TEST - And function to flags, no result:				
POPF-Pop flags	1 0 0 1 1 1 0 1				Register/memory and register	1 0 0 0 0 1 0 w	mod reg	r/m	
					Immediate data and register/memory	1 1 1 1 0 1 1 w	mod 0 0 0 r/m	data	data if w=1
					Immediate data and accumulator	1 0 1 0 1 0 0 w	data	data if w=1	
Arithmetic					OR - Or:				
ADD - Add:					Reg./memory and register to either	0 0 0 0 1 0 d w	mod reg	r/m	
Reg./memory with register to either	0 0 0 0 0 0 d w	mod reg	r/m		Immediate to register/memory	1 0 0 0 0 0 w	mod 0 0 1 r/m	data	data if w=1
Immediate to register/memory	1 0 0 0 0 0 s w	mod 0 0 0 r/m	data	data if s/w=01	Immediate to accumulator	0 0 0 0 1 1 0 w	data	data if w=1	
Immediate to accumulator	0 0 0 0 0 1 0 w	data	data if w=1						
ADC - Add with carry:					XOR - Exclusive or:				
Reg./memory with register to either	0 0 0 1 0 0 d w	mod reg	r/m		Reg./memory and register to either	0 0 1 1 0 0 d w	mod reg	r/m	
Immediate to register/memory	1 0 0 0 0 0 s w	mod 0 1 0 r/m	data	data if s/w=01	Immediate to register/memory	1 0 0 0 0 0 w	mod 1 1 0 r/m	data	data if w=1
Immediate to accumulator	0 0 0 1 0 1 0 w	data	data if w=1		Immediate to accumulator	0 0 1 1 0 1 0 w	data	data if w=1	
INC - Increment:									
Register/memory	1 1 1 1 1 1 1 w	mod 0 0 0 r/m			String Manipulation				
Register	0 1 0 0 0 reg				REP-Repeat	1 1 1 1 0 0 1 2			
AAA-ASCII adjust for add	0 0 1 1 0 1 1 1				MOVSB=Move byte/word	1 1 0 1 0 0 1 w			
DAA-Decimal adjust for add	0 0 1 0 0 1 1 1				CMPSB=Compare byte/word	1 1 0 0 0 1 1 w			
					SCASB=Scan byte/word	1 1 0 1 0 1 1 w			
SUB - Subtract:					LDSB=Load byte/word to AL/AX	1 1 0 1 0 1 0 w			
Reg./memory and register to either	0 0 1 0 1 0 d w	mod reg	r/m		STOSB=Store byte/word from AL/AX	1 1 0 1 0 1 0 w			
Immediate to register/memory	1 0 0 0 0 0 s w	mod 1 0 1 r/m	data	data if s/w=01					
Immediate from accumulator	0 0 1 0 1 1 0 w	data	data if w=1						
SBB - Subtract with borrow					Control Transfer				
Reg./memory and register to either	0 0 0 1 1 0 d w	mod reg	r/m		CALL - Call:				
Immediate from register/memory	1 0 0 0 0 0 s w	mod 0 1 1 r/m	data	data if s/w=01	Direct within segment	1 1 1 1 0 1 0 0 0	disp-low	disp-high	
Immediate from accumulator	0 0 0 1 1 1 0 w	data	data if w=1		Indirect within segment	1 1 1 1 1 1 1 1	mod 0 1 0 r/m		
DEC - Decrement:					Direct intersegment	1 1 0 0 1 0 1 0	offset-low	offset-high	
Register/memory	1 1 1 1 1 1 1 w	mod 0 0 1 r/m					seg-low	seg-high	
Register	0 1 0 0 1 reg				Indirect intersegment	1 1 1 1 1 1 1 1	mod 0 1 1 r/m		
NEB-Change sign	1 1 1 1 0 1 1 w	mod 0 1 1 r/m							

continued

Table 4. 8086 Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code	Mnemonic and Description	Instruction Code
JMP - Unconditional Jump:	7 8 5 4 3 2 1 0 7 8 5 4 3 2 1 0 7 8 5 4 3 2 1 0	JNS Jump on not sign	7 8 5 4 3 2 1 0 7 8 5 4 3 2 1 0
Direct within segment	1 1 1 0 1 0 0 1 disp-low disp-high		0 1 1 1 1 0 0 1 disp
Direct within segment-short	1 1 1 0 1 0 1 1 disp	LOOP Loop CX times	1 1 1 0 0 0 1 0 disp
Indirect within segment	1 1 1 1 1 1 1 1 mod 1 0 0 r/m	LOOPZ/LOOPE Loop while zero/equal	1 1 1 0 0 0 0 1 disp
Direct intersegment	1 1 1 0 1 0 1 0 offset-low offset-high	LOOPNZ/LOOPNE Loop while not zero/equal	1 1 1 0 0 0 0 0 disp
	seg-low seg-high	JCZ Jump on CX zero	1 1 1 0 0 0 1 1 disp
Indirect intersegment	1 1 1 1 1 1 1 1 mod 1 0 1 r/m		
RET - Return from CALL:		INT Interrupt	
Within segment	1 1 0 0 0 0 1 1	Type specified	1 1 0 0 1 1 0 1 type
Within seg adding immed to SP	1 1 0 0 0 0 1 0 data-low data-high	Type 3	1 1 0 0 1 1 0 0
Intersegment	1 1 0 0 1 0 1 1	INT0 Interrupt on overflow	1 1 0 0 1 1 1 0
Intersegment, adding immediate to SP	1 1 0 0 1 0 1 0 data-low data-high	INTE Interrupt return	1 1 0 0 1 1 1 1
JE/JZ Jump on equal/zero	0 1 1 1 0 1 0 0 disp	Processor Control	
JL/JNGE Jump on less/not greater or equal	0 1 1 1 1 1 0 0 disp	CLC Clear carry	1 1 1 1 1 0 0 0
JLE/JNG Jump on less or equal/not greater	0 1 1 1 1 1 1 0 disp	CMC Complement carry	1 1 1 1 0 1 0 1
JNB/JNAE Jump on below/not above or equal	0 1 1 1 0 0 1 0 disp	STC Set carry	1 1 1 1 1 0 0 1
JNB/JNA Jump on below or equal/not above	0 1 1 1 0 1 1 0 disp	CLD Clear direction	1 1 1 1 1 1 0 0
JPF/JPE Jump on parity/parity even	0 1 1 1 1 0 1 0 disp	STD Set direction	1 1 1 1 1 1 0 1
JO Jump on overflow	0 1 1 1 0 0 0 0 disp	CLI Clear interrupt	1 1 1 1 1 0 1 0
JS Jump on sign	0 1 1 1 1 0 0 0 disp	STI Set interrupt	1 1 1 1 1 0 1 1
JNE/JNZ Jump on not equal/not zero	0 1 1 1 0 1 0 1 disp	HLT Halt	1 1 1 1 1 0 1 0
JNL/JNG Jump on not less/greater or equal	0 1 1 1 1 1 0 1 disp	WAIT Wait	1 0 0 1 1 0 1 1
JNLE/JB Jump on not less or equal/greater	0 1 1 1 1 1 1 1 disp	ESC Escape (to external device)	1 1 0 1 1 x x x mod x x x r/m
JNB/JAE Jump on not below/above or equal	0 1 1 1 0 0 1 1 disp	LOCK Bus lock prefix	1 1 1 1 0 0 0 0
JNBE/JA Jump on not below or equal/above	0 1 1 1 0 1 1 1 disp		
JNP/JPO Jump on not par/par odd	0 1 1 1 1 0 1 1 disp		
JNO Jump on not overflow	0 1 1 1 0 0 0 1 disp		
Notes		if s:w = 01 then 16 bits of immediate data form the operand.	
AL = 8-bit accumulator		if s:w = 11 then an immediate data byte is sign extended to form the 16-bit operand.	
AX = 16-bit accumulator		if v = 0 then "count" = 1; if v = 0 then "count" in (CL)	
CX = Count register		x = don't care.	
DS = Data segment		if v = 0 then "count" = 1; if v = 1 then "count" in (CL) register.	
ES = Extra segment		z is used for string primitives for comparison with ZF FLAG.	
Above/below refers to unsigned value.		SEGMENT OVERRIDE PREFIX	
Greater = more positive;		0 0 1 reg 1 1 0	
Less = less positive (more negative) signed values		REG is assigned according to the following table:	
if d = 1 then "to" reg; if d = 0 then "from" reg			
if w = 1 then word instruction; if w = 0 then byte instruction			
if mod = 11 then r/m is treated as a REG field			
if mod = 00 then DISP = 0*, disp-low and disp-high are absent			
if mod = 01 then DISP = disp-low sign-extended to 16-bits, disp-high is absent			
if mod = 10 then DISP = disp-high; disp-low			
if r/m = 000 then EA = (BX) + (SI) + DISP			
if r/m = 001 then EA = (BX) + (DI) + DISP			
if r/m = 010 then EA = (BP) + (SI) + DISP			
if r/m = 011 then EA = (BP) + (DI) + DISP			
if r/m = 100 then EA = (SI) + DISP			
if r/m = 101 then EA = (DI) + DISP			
if r/m = 110 then EA = (BP) + DISP*			
if r/m = 111 then EA = (BX) + DISP			
DISP follows 2nd byte of instruction (before data if required)			
*except if mod = 00 and r/m = 110 then EA = disp-high; disp-low.			
		Instructions which reference the flag register file as a 16-bit object use the symbol FLAGS to represent the file:	
		FLAGS = X:X:X:(OF):(DF):(IF):(TF):(SF):(ZF):(AF):X:(PF):X:(CF)	
		Mnemonics < Intel, 1978	

SPECIFICATIONS

Central Processor

CPU — 8086 (5 MHz clock rate)

Note

May be operated at 2.5 MHz or 5 MHz, jumper selectable, for use with 8086.

Memory

ROM — 8K bytes 2316/2716

RAM — 2K bytes (expandable to 4K bytes) 2142

Addressing

ROM — FE000-FFFF

RAM — 0-7FF (800-FFF available with additional 2142's)

Note

The wire-wrap area of the SDK-86 PC board may be used for additional custom memory expansion.

Input/Output

Parallel — 48 lines (two 8255A's)

Serial — RS232 or current loop (8251A)

Baud Rate — selectable from 110 to 4800 baud

SDK-86

Interfaces

Bus — All signals TTL compatible

Parallel I/O — All signals TTL compatible

Serial I/O — 20 mA current loop TTY or RS232

Note

The user has access to all bus signals which enable him to design custom system expansions into the kit's wire-wrap area.

Interrupts (256 vectored)

Maskable

Non-maskable

TRAP

DMA

Hold Request — Jumper selectable. TTL compatible input.

Software

System Monitor — Preprogrammed 2716 or 2316 ROMs

Addresses — FE000-FFFF

Monitor I/O — Keyboard/display or TTY or CRT (serial I/O)

Physical Characteristics

Width — 13.5 in. (34.3 cm)

Height — 12 in. (30.5 cm)

Depth — 1.75 in. (4.45 cm)

Weight — approx. 24 oz. (3.3 kg)

Electrical Characteristics

DC Power Requirement

(Power supply not included in kit)

Voltage	Current
$V_{CC} 5V \pm 5\%$	3.5A
$V_{TTY} - 12V \pm 10\%$	0.3A
(V _{TTY} required only if teletype is connected)	

Environmental Characteristics

Operating Temperature — 0-50°C

Reference Manuals

9800697A — SDK-86 MCS-86 System Design Kit Assembly Manual

9800722 — MCS-86 User's Manual

9800640A — 8086 Assembly Language Programming Manual

8086 Assembly Language Reference Card

Reference manuals are shipped with each product only if designated SUPPLIED (see above). Manuals may be ordered from any Intel sales representative, distributor office or from Intel Literature Department, 3065 Bowers Avenue, Santa Clara, California 95051.

ORDERING INFORMATION

Part Number	Description
SDK-86	MCS-86 system design kit